

## PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2004-259106

(43)Date of publication of application : 16.09.2004

---

(51)Int.Cl.

G06F 9/46

---

(21)Application number : 2003-  
050701

(71)Applicant : NTT DATA CORP

(22)Date of filing :

27.02.2003

(72)Inventor : TANIGUCHI HIDEO

NOMURA YOSHINARI

TANAKA KAZUO

ITO KENICHI

NAKAJIMA YUSAKU

TABUCHI MASAKI

MASUMOTO KEI

---

(54) MULTI-OPERATING SYSTEM CONTROL METHODPROGRAM FOR  
MAKING COMPUTER PERFORM METHODAND MULTI-OPERATING SYSTEM  
CONTROL DEVICE

(57)Abstract:

PROBLEM TO BE SOLVED: To provide a multi-operating system control method capable of improving the security and reliability of a multi-operating systemand provide a program for making a computer perform the method and a multi-operating system control device.

SOLUTION: In the case of transmitting data from OS201 to OS202a register data read/write processing part 24 of OS201 writes transmit data in OS-to-OS data

communication register 18. After switching from OS201 to OS202 is performed by an OS switching processing part 281a register data read/write processing part 242 of the OS202 reads out data from the OS-to-OS data communication register 18.

---

## CLAIMS

---

[Claim(s)]

[Claim 1]

It is the multi operating system control method which carries out switching control of the 1st operating system and 2nd operating system that work by one computer Like data storage Takumi who stores data which becomes a predetermined register with a communication object during execution of said 1st operating system

A data acquisition process of acquiring data which serves as said communication object from said predetermined register when a change is made by said 2nd operating system from said 1st operating system

it is \*\*\*\*\* -- a multi operating system control method characterized by things.

[Claim 2]

When data which becomes said predetermined register with a communication object like said data storage Takumi is stored. Virtual memory was beforehand carried out to logical address N of the 1st virtual memory space corresponding to said 1st operating system. With the 1st change command changed from said 1st operating system to said 2nd operating system. From said 1st operating system change to said 2nd operating system and by said data acquisition process.

A data acquisition command memorized beforehand in the logical address N+1 of the 2nd virtual memory space corresponding to said 2nd operating system Or a multi operating system control method according to claim 1 executing a data acquisition command executed after a command memorized beforehand in this logical address N+1 and acquiring data which serves as said communication

object from said predetermined register.

[Claim 3]

After said 2nd operating system executes said data acquisition commandWith the 2nd change command for changing from said 2nd operating system by which virtual memory was carried out beforehand to said 1st operating system. A multi operating system control method according to claim 2 changing from said 2nd operating system to said 1st operating system.

[Claim 4]

A data backup process of backing up data stored in a predetermined registerSaid backed-up dataincluding further a data restoration process restored to said predetermined register in which this data was stored like said data storage TakumiStore data used as a communication object in a register in which data was backed up by said data backup processand with said 2nd change command. When a change is performed to said 1st operating system from said 2nd operating systemby said data restoration process. A multi operating system control method according to claim 3 restoring said backed-up data to said predetermined register in which this data was stored.

[Claim 5]

A multi operating system control method of a statement of even either of claims 1-4wherein said predetermined register is a register which is not used by the 1st operating system during execution of said 1st operating system.

[Claim 6]

A multi operating system control method according to claim 5wherein said predetermined register is a debugging register and/or a general register which are not used by the 1st operating system during execution of said 1st operating system.

[Claim 7]

Like said data storage Takumisize of data used as said communication object includes further a data size determination process which judges whether it is size storable in said predetermined registerWhen it can storestore data in a register

predetermined [ this ] and when it cannot store A multi operating system control method of a statement of even either of claims 1-6 wherein it stores data which serves as said communication object by storing means other than said predetermined register and said 2nd operating system acquires stored data.

[Claim 8]

Said storing means is a storing means with which a Network Interface Card was equipped When data used as said communication object addressed to said 2nd operating system is stored in this storing means A multi operating system control method according to claim 7 relaying said 1st operating system and setting connection of this storing means as said 2nd operating system.

[Claim 9]

A program which makes a computer perform a multi operating system control method that even either of claims 1-8 was indicated.

[Claim 10]

It is a multi operating system control device which carries out switching control of the 1st operating system and 2nd operating system that work by one computer A data storing means which stores data which becomes a predetermined register with a communication object during execution of said 1st operating system

A data acquisition means which acquires data which serves as said communication object from said predetermined register when a change is made by said 2nd operating system from said 1st operating system

A multi operating system control device characterized by preparation \*\*\*\*\*.

[Claim 11]

When data which becomes said predetermined register with a communication object by said data storing means is stored. Virtual memory was beforehand carried out to logical address N of the 1st virtual memory space corresponding to said 1st operating system. With the 1st change command changed from said 1st operating system to said 2nd operating system. From said 1st operating system change to said 2nd operating system and said data acquisition means A data acquisition command memorized beforehand in the logical address N+1 of

the 2nd virtual memory space corresponding to said 2nd operating system Or the multi operating system control device according to claim 10 executing a data acquisition command executed after a command memorized beforehand in this logical address N+1 and acquiring data which serves as said communication object from said predetermined register.

---

## DETAILED DESCRIPTION

---

[Detailed Description of the Invention]

[0001]

[Field of the Invention]

This invention relates to the multi operating system control method for carrying out switching control of two or more operating systems which work on one computer the program which makes a computer perform the method and a multi operating system control device.

In particular it is related with security the multi operating system control method which can raise reliability the program which makes a computer perform the method and an operating system control device.

[0002]

[Description of the Prior Art]

In the usual computer one operating system operates it manages computer resource such as a processor of a computer a memory and a secondary memory and the resource schedule is carried out so that the computer can operate efficiently. There are various kinds of operating systems. It is various such as what is excellent in batch processing what is excellent in TSS (Time Sharing System) a thing excellent in GUI (Graphical User Interface).

[0003]

On the other hand there are needs to perform simultaneously these two or more

\*\*\*\* operating system by one computer. For example in a mainframe there is a demand that the operating system which performs on-line processing accompanying actual business and the operating system for development want to operate by one computer. Or there is also a demand that the operating system with which GUI is ready and an operating system excellent in real-time requirement want to work simultaneously etc.

[0004]

However each operating system is designed considering managing a computer resource independently as a premise. Therefore coexistence of two or more operating systems is impossible without a certain mechanism.

[0005]

As a mechanism in which two or more operating systems are operated on one computer there is a virtual-machine method realized with the mainframe. Drawing 21 is a block diagram showing the example 1 of composition of the conventional multi operating system by the above-mentioned virtual-machine method.

[0006]

The multi operating system shown in the figure comprises hardware 1 and base OS 2 the VM (Virtual Machine) monitor 3 virtual OS (operating system) 4<sub>1</sub> virtual OS 4<sub>2</sub> and virtual OS 4<sub>3</sub>.

[0007]

The hardware 1 are CPU (Central Processing Unit) a physical memory input/output device etc. Base OS 2 and the VM monitor 3 are controlling all the hardware 1. The VM monitor 3 emulates the interface to each hardware 1 of virtual OS 4<sub>1</sub> - 4<sub>3</sub>. Virtual OS 4<sub>1</sub> - 4<sub>3</sub> run on base OS 2. That is base OS 2 and virtual OS 4<sub>1</sub> - 4<sub>3</sub> are in a master-slave relation.

[0008]

In the multi operating system shown in the figure when communicating data between virtual OS's the method of going via the shared memory (graphic display abbreviation) which can be referred to simultaneously from base OS 2 and virtual OS 4<sub>1</sub> - 4<sub>3</sub> is taken.

[0009]

For example in transmitting data to virtual OS<sub>2</sub> from virtual OS<sub>1</sub> after virtual OS<sub>1</sub> stores data in a shared memory virtual OS<sub>2</sub> acquires data from a shared memory. Communication is performed via shared memory and base OS between virtual OS and other devices in addition to between virtual OS's.

[0010]

There is a microkernel method as art of providing the interface of two or more operating systems by one computer. Drawing 22 is a block diagram showing the example 2 of composition of the conventional multi operating system by the above-mentioned microkernel method.

[0011]

The multi operating system shown in the figure comprises the hardware 5 the microkernel (control program) 6 OS<sub>1</sub> OS<sub>2</sub> and OS<sub>3</sub>.

[0012]

In the microkernel method OS<sub>1</sub> which provides the operating system function shown to a user on the microkernel 6 - 7<sub>3</sub> are built. A user uses the hardware 5 (computer resource) via OS<sub>1</sub> - 7<sub>3</sub>. The microkernel 6 controls OS<sub>1</sub> - 7<sub>3</sub>.

[0013]

In the multi operating system shown in the figure also when communicating data between OS's a microkernel and the method of going via the shared memory (graphic display abbreviation) which can be referred to simultaneously from OS<sub>1</sub> - 7<sub>3</sub> are taken.

[0014]

For example in transmitting data to OS<sub>2</sub> from OS<sub>1</sub> after OS<sub>1</sub> stores data in a shared memory OS<sub>2</sub> acquires data from a shared memory. Communication is performed via a shared memory or the microkernel 6 between OS and other devices in addition to between OS's.

[0015]

[Patent documents 1]

JP11-149385A

[Patent documents 2]

JP2001-216172A

[Patent documents 3]

JP2000-207232A

[Patent documents 4]

JP8-212089A

[Patent documents 5]

JP2001-290661A

[Patent documents 6]

JP11-85546A

[Patent documents 7]

JP2001-282558A

[Patent documents 8]

JP11-85546A

[0016]

[Problem(s) to be Solved by the Invention]

By the way, since it was communicating via the shared memory between virtual OS's and between virtual OS and other devices in the conventional multi operating system shown in drawing 21 as mentioned above, there was a problem that security was low.

[0017]

That is, since it can be referred to simultaneously when it is accessed unlawfully to the above-mentioned shared memory by the shared memory from the outside at the time of communication from each of two or more virtual OS<sub>1</sub> - 4<sub>3</sub>, virtual OS<sub>1</sub> - 4<sub>3</sub> are attacked simultaneously.

[0018]

Also in the conventional multi operating system shown in drawing 22a, shared memory simultaneously from each of two or more OS<sub>1</sub> - 7<sub>3</sub>. Since it can be referred to when it is accessed unlawfully by the shared memory from the outside at the time of communication, OS<sub>1</sub> - 7<sub>3</sub> are attacked simultaneously and there is

a security top problem.

[0019]

This invention is made in view of the above and is a thing.

The purpose is to provide the security of the multi operating system control method which can raise reliability of the program which makes a computer perform the method and a multi operating system control device.

[0020]

[Means for Solving the Problem]

To achieve the above objects, if it is in a multi operating system control method concerning this invention, it is the multi operating system control method which carries out switching control of the 1st operating system and 2nd operating system that work by one computer. Like data storage Takumi who stores data which becomes a predetermined register with a communication object during execution of said 1st operating system, when a change was made by said 2nd operating system from said 1st operating system, a data acquisition process of acquiring data which serves as said communication object from said predetermined register was included.

[0021]

According to this invention, data which becomes a predetermined register with a communication object is stored during execution of the 1st operating system. Since we decided to acquire data which serves as a communication object from a predetermined register when a change was made by the 2nd operating system from the 1st operating system, by using for communication between operating systems a register in which size of storable data was restricted without passing a shared memory, the possibility of abnormal operation of each operating system by communication of inaccurate data can be avoided and security of a multi operating system and reliability can be raised.

[0022]

If it is in a multi operating system control method concerning the next

inventionWhen data which becomes said predetermined register with a communication object like said data storage Takumi is stored. Virtual memory was beforehand carried out to logical address N of the 1st virtual memory space corresponding to said 1st operating system. With the 1st change command changed from said 1st operating system to said 2nd operating system. From said 1st operating systemchange to said 2nd operating system and by said data acquisition process. A data acquisition command memorized beforehand in the logical address N+1 of the 2nd virtual memory space corresponding to said 2nd operating systemOr a data acquisition command executed after a command memorized beforehand in this logical address N+1 is executedand data which serves as said communication object from said predetermined register is acquired.

[0023]

When data used as a communication object is stored in a predetermined register according to this invention. Virtual memory was beforehand carried out to logical address N of the 1st virtual memory space corresponding to the 1st operating system. With the 1st change command changed from the 1st operating system to the 2nd operating system. It changes from the 1st operating system to the 2nd operating systemA data acquisition command memorized beforehand in the logical address N+1 of the 2nd virtual memory space corresponding to the 2nd operating systemOr since a data acquisition command executed after a command memorized beforehand in this logical address N+1 is executed and data which serves as a communication object from a predetermined register is acquiredIntersectionssuch as the conventional base OS and VM monitorcan be unnecessarythe 1st and 2nd operating systems can live togetherand security of a multi operating system and reliability can be raised.

[0024]

If it is in a multi operating system control method concerning the next inventionAfter said 2nd operating system executes said data acquisition commandWith the 2nd change command for changing from said 2nd operating

system by which virtual memory was carried out beforehand to said 1st operating system it changes from said 2nd operating system to said 1st operating system.

[0025]

According to this invention the 2nd operating system With the 2nd change command for changing from the 2nd operating system by which virtual memory was carried out beforehand to the 1st operating system after executing a data acquisition command. A change and return of an operating system can be performed smoothly maintaining high security and reliability since it changes from the 2nd operating system to the 1st operating system.

[0026]

If it is in a multi operating system control method concerning the next invention A data backup process of backing up data stored in a predetermined register Said backed-up data including further a data restoration process restored to said predetermined register in which this data was stored like said data storage TakumiStore data used as a communication object in a register in which data was backed up by said data backup process and with said 2nd change command. When a change is performed to said 1st operating system from said 2nd operating system said data backed up by said data restoration process is restored to said predetermined register in which this data was stored.

[0027]

According to this invention store data used as a communication object in a register which backs up data stored in a predetermined register and by which data was backed up and with the 2nd change command. Since we decided to restore backed-up data to a predetermined register in which data was stored when a change was performed to the 1st operating system from the 2nd operating system By being able to store data which communicates between operating systems in a register currently used during execution of the 1st operating system and communicating using the register Security of a multi operating system and reliability can be raised.

[0028]

If it is in a multi operating system control method concerning the next inventionsaid predetermined register is characterized by being a register which is not used by the 1st operating system during execution of said 1st operating system.

[0029]

Since it presupposed that it is a predetermined register a register which is not used by the 1st operating system during execution of the 1st operating system according to this inventionData which communicates between operating systems can be stored and security of a multi operating system and reliability can be raised by communicating using the register.

[0030]

If it is in a multi operating system control method concerning the next inventionSaid predetermined register is characterized by being a debugging register and/or a general register which are not used by the 1st operating system during execution of said 1st operating system.

[0031]

Since it presupposed that it is a predetermined register a debugging register and/or a general register which are not used by the 1st operating system during execution of the 1st operating system according to this inventionData which communicates between operating systems can be stored in a debugging register and/or a general registerand security of a multi operating system and reliability can be raised by communicating using the register.

[0032]

If it is in a multi operating system control method concerning the next inventionLike said data storage Takumisize of data used as said communication object includes further a data size determination process which judges whether it is size storable in said predetermined registerWhen it can storedata is stored in a register predetermined [ this ]when it cannot storedata which serves as said communication object by storing means other than said predetermined register is storedand said 2nd operating system acquires stored data.

[0033]

According to this invention, size of data used as a communication object judges whether it is size storable in a predetermined register. When it can store data in a predetermined register, and when it cannot store. Since data which serves as a communication object by storing means other than a predetermined register is stored, and the 2nd operating system acquires stored data. A suitable storing means can be chosen according to size of data, and data between efficient operating systems which have high security and reliability can be communicated.

[0034]

If it is in a multi operating system control method concerning the next invention, a Network Interface Card is equipped with said storing means. When data used as said communication object addressed to said 2nd operating system is stored in this storing means, said 1st operating system is relayed, and connection of this storing means is set as said 2nd operating system.

[0035]

When data used as said communication object addressed to the 2nd operating system is stored in a storing means with which a Network Interface Card was equipped according to this invention, since the 1st operating system is relayed, and connection of a storing means is set as the 2nd operating system. Even if size of data used as a communication object is size unstorable in a predetermined register, data between efficient operating systems which have high security and reliability can be communicated.

[0036]

A program concerning the next invention is a program which makes a computer perform a multi operating system control method that even either of the above-mentioned inventions was indicated. Computer reading of the program becomes possible, and any one operation of the above-mentioned invention by this can be performed by computer.

[0037]

If it is in a multi operating system control device concerning the next invention, it is

a multi operating system control device which carries out switching control of the 1st operating system and 2nd operating system that work by one computerA data storing means which stores data which becomes a predetermined register with a communication object during execution of said 1st operating systemWhen a change was made by said 2nd operating system from said 1st operating systemit had a data acquisition means which acquires data which serves as said communication object from said predetermined register.

[0038]

According to this inventiondata which becomes a predetermined register with a communication object is stored during execution of the 1st operating systemSince we decided to acquire data which serves as a communication object from a predetermined register when a change was made by the 2nd operating system from the 1st operating systemBy using for communication between operating systems a register in which size of storable data was restrictedwithout passing a shared memory. The possibility of abnormal operation of each operating system by communication of inaccurate data can be avoidedand security of a multi operating system and reliability can be raised.

[0039]

If it is in a multi operating system control device concerning the next inventionWhen data which becomes said predetermined register with a communication object by said data storing means is stored. Virtual memory was beforehand carried out to logical address N of the 1st virtual memory space corresponding to said 1st operating system. With the 1st change command changed from said 1st operating system to said 2nd operating system. From said 1st operating systemchange to said 2nd operating system and said data acquisition meansA data acquisition command memorized beforehand in the logical address N+1 of the 2nd virtual memory space corresponding to said 2nd operating systemOr a data acquisition command executed after a command memorized beforehand in this logical address N+1 is executedand data which serves as said communication object from said predetermined register is

acquired.

[0040]

When data used as a communication object is stored in a predetermined register according to this invention. Virtual memory was beforehand carried out to logical address N of the 1st virtual memory space corresponding to the 1st operating system. With the 1st change command changed from the 1st operating system to the 2nd operating system. It changes from the 1st operating system to the 2nd operating system. A data acquisition command memorized beforehand in the logical address N+1 of the 2nd virtual memory space corresponding to the 2nd operating system. Or since a data acquisition command executed after a command memorized beforehand in this logical address N+1 is executed and data which serves as a communication object from a predetermined register is acquired. Intersectionssuch as the conventional base OS and VM monitor can be unnecessary. the 1st and 2nd operating systems can live together and security of a multi operating system and reliability can be raised.

[0041]

[Embodiment of the Invention]

Hereafter Embodiments 1 and 2 of the multi operating system control method which starts this invention with reference to drawings. the program which makes a computer perform the method and a multi operating system control device are described in detail.

[0042]

(Embodiment 1)

Drawing 1 is a block diagram showing the outline composition of Embodiment 1 concerning this invention. The multi operating system shown in this figure comprises the hardware 10, OS<sub>1</sub>, OS<sub>2</sub> and AP(application program) 30. 1 - 30

4.

[0043]

OS<sub>1</sub> and OS<sub>2</sub> have register data read-and-write treating part 24<sub>1</sub> and 24<sub>2</sub>. OS spawn process part 28<sub>1</sub> and 28<sub>2</sub> respectively. And the hardware 10 has the

register 18 for the data communications between OS's.

[0044]

The register 18 for the data communications between OS's from OS<sub>2 1</sub> to OS<sub>2 2</sub>. Or when communicating data from OS<sub>2 2</sub> to OS<sub>2 1</sub> it is a register which stores the data which communicates and is a register which is not used by OS<sub>2 1</sub> or OS<sub>2 2</sub> under execution. The command over OS of a transmission destination etc. are included in the data which communicates.

[0045]

For example except the time of a debug mode the debugging register of the Pentium (registered trademark) processor by Intel is a register which is not used and can be used as the register 18 for the data communications between OS's.

[0046]

Similarly it is used in order to count the number of times of a loop but the general register ECX of the Pentium (registered trademark) processor by Intel is not used when loop processing is not being performed but it can be used as the register 18 for the data communications between OS's.

[0047]

The register which can be used as the register 18 for the data communications between OS's should just be a register which is not limited to the above-mentioned register and is not used by OS<sub>2 1</sub> or OS<sub>2 2</sub> under execution.

[0048]

If the data stored in the register is backed up in a memory etc. even if it is the register currently used by OS<sub>2 1</sub> or OS<sub>2 2</sub> under execution the register can be used as the register 18 for the data communications between OS's.

[0049]

However the backed-up data is restored to a register after the end of communication of data in this case. By doing in this way access to the register after the end of communication of data can be normally performed now like the state before performing data communications.

[0050]

In the multi operating system the change to OS2<sub>2</sub> from OS2<sub>1</sub> or OS2<sub>1</sub> from OS2<sub>2</sub> is performed by OS spawn process part 28<sub>1</sub> and 28<sub>2</sub> respectively. As a generation factor of interruption the needed information of the data through the register 18 for the data communications between OS's the periodical change demand by a timer (graphic display abbreviation) etc. are mentioned.

[0051]

In the interrupt by the needed information of data occurring and transmitting data to OS2<sub>2</sub> from OS2<sub>1</sub> The data transmitted to the register 18 for the data communications between OS's by register data read-and-write treating part 24<sub>1</sub> of OS2<sub>1</sub> is written in (stored).

[0052]

And after the change to OS2<sub>2</sub> from OS2<sub>1</sub> was performed by OS spawn process part 28<sub>1</sub> Data is read from the register 18 for the data communications between OS's by register data read-and-write treating part 24<sub>2</sub> of OS2<sub>2</sub> (acquired) and the data communications between OS's are completed.

[0053]

When transmitting data to OS2<sub>1</sub> from OS2<sub>2</sub> the data transmitted to the register 18 for the data communications between OS's by register data read-and-write treating part 24<sub>2</sub> of OS2<sub>2</sub> is written in. And after the change to OS2<sub>1</sub> from OS2<sub>2</sub> is performed by OS spawn process part 28<sub>2</sub> data is read from the register 18 for the data communications between OS's by register data read-and-write treating part 24<sub>1</sub> of OS2<sub>1</sub>.

[0054]

Here since the size of the register 18 for the data communications between OS's is limited with 16 bits - about 80 bits neither inaccurate data nor a virus program can spread it easily between OS2<sub>1</sub> and OS2<sub>2</sub> and it can raise the security and reliability of OS.

[0055]

Thus the multi operating system The feature is at a common control program and

the point of realizing delivery of data between OS20<sub>1</sub> and OS20<sub>2</sub> using the register 18 for the data communications between OS's without passing a shared memory etc. where high security is held like before.

[0056]

Drawing 2 is a block diagram showing the concrete composition of Embodiment 1. The same numerals are attached to the portion corresponding to each part of drawing 1 in this figure. This Embodiment 1 explains the case where the debugging register 15 of the Pentium (registered trademark) processor by Intel is used as the register 18 for the data communications between OS's.

[0057]

In drawing 2 the hardware 10 is a computer resource and The control section 11 the physical memory 12 the interrupt vector table register 13 the page table register 14 the debugging register 15 the other registers 16 the program counter 17 and the keyboard that is not illustrated It comprises a display etc.

[0058]

OS20<sub>1</sub> and OS20<sub>2</sub> of multi-composition exist in the hardware 10. Each of OS20<sub>1</sub> and OS20<sub>2</sub> manages the computer resource of the hardware 10 and is changed in an interrupt handler. That is considering a certain time the hardware 10 is occupied by one OS (OS20<sub>1</sub> or OS20<sub>2</sub>).

[0059]

OS20<sub>1</sub> Data check treating part 21, interruption distribution treating part 22, It comprises interrupt processing section 23, register data read-and-write treating part 24, interrupt vector table 25, page table 26, register save area 27, etc.

[0060]

OS change treating part 28, shown in drawing 1 It corresponds to data check treating part 21, interruption distribution treating part 22, interrupt processing section 23, interrupt vector table 25, page table 26, register save area 27, etc. AP30<sub>1</sub> and AP30<sub>2</sub> are application programs which run on OS20<sub>1</sub>.

[0061]

On the other hand OS20<sub>2</sub> is put side by side to OS20<sub>1</sub> Data check treating part

21 2 interruption distribution treating part 22 2 It comprises interrupt processing section 23 2 register data read-and-write treating part 24 2 interrupt vector table 25 2 page table 26 2 register save area 27 2 etc.

[0062]

OS change treating part 28 2 shown in drawing 1 It corresponds to data check treating part 21 2 interruption distribution treating part 22 2 interrupt processing section 23 2 interrupt vector table 25 2 page table 26 2 register save area 27 2 etc. AP30 3 and AP30 4 are application programs which run on OS20 2.

[0063]

In the hardware 10 the control section 11 is CPU etc. and controls each part by program execution. The physical memory 12 is mass memory storage which exists really physically as shown in drawing 3 is used for a virtual-storage-supervision method and actually memorizes various commands data etc.

[0064]

That is in this Embodiment 1 virtual-memory-space 40 1 and virtual-memory-space 40 2 are virtually built corresponding to the physical memory 12. Virtual-memory-space 40 1 is provided corresponding to the OS20 1 side and is mapped by physical memory field 12a1 for OS of the physical memory 12 and physical memory field 12b1 for AP.

[0065]

Virtual-memory-space 40a1 for OS is memory space used by OS20 1 and specifically is mapped by physical memory field 12a1 for OS. Virtual-memory-space 40b1 for AP is memory space used by AP30 1 and AP30 2 and is mapped by physical memory field 12b1 for AP.

[0066]

On the other hand virtual-memory-space 40 2 is provided corresponding to the OS20 2 side and is mapped by physical memory field 12a2 for OS of the physical memory 12 and physical memory field 12b2 for AP.

[0067]

Virtual-memory-space 40a2 for OS is memory space used by OS20

and specifically is mapped by physical memory field 12a<sub>2</sub> for OS. Virtual-memory-space 40b<sub>2</sub> for AP is memory space used by AP30<sub>3</sub> and AP30<sub>4</sub> and is mapped by physical memory field 12b<sub>2</sub> for AP.

[0068]

The physical address is given to the physical storage unit in the physical memory 12. On the other hand in virtual-memory-space 40<sub>1</sub> and virtual-memory-space 40<sub>2a</sub> a command and data are specified by the logical address used in a program. In the virtual-storage-supervision method the logical address is changed into the physical address with a paging system segmentation and paging/segmentation.

[0069]

Address translation is performed in a paging system by the block unit (4 K bytes) called a page. For this reason in OS20<sub>1</sub> page table (translation table) 26<sub>1</sub> showing whether the virtual page of virtual-memory-space 40<sub>1</sub> supports the physical page of physical memory 12 throat is provided. Similarly also in OS20<sub>2</sub> page table (translation table) 26<sub>2</sub> showing whether the page of virtual-memory-space 40<sub>2</sub> is equivalent to the page of physical memory 12 throat is provided.

[0070]

As shown in drawing 4 (b) the data write command A which writes the data which communicates from OS20<sub>1</sub> to OS20<sub>2</sub> in the debugging register 15 is memorized by Z address (logical address) of virtual-memory-space 40<sub>1</sub>. The change command A for changing from OS20<sub>1</sub> to OS20<sub>2</sub> is memorized by N address of virtual-memory-space 40<sub>1</sub>.

[0071]

On the other hand in virtual-memory-space 40<sub>2</sub> the change command B for changing from OS20<sub>2</sub> to OS20<sub>1</sub> is memorized by the same N address as N address of virtual-memory-space 40<sub>1</sub>. Data read instruction B which reads the data written in the debugging register 15 to the N+1st street of virtual-memory-space 40<sub>2</sub> is memorized.

[0072]

Returning to drawing 2 the interrupt vector table register 13 is a register which

stores the interruption number according to interruption. This interruption number supports the logical address of the interrupt handler in interrupt vector table 25<sub>1</sub> or interrupt vector table 25<sub>2</sub> mentioned later.

[0073]

The page table register 14 is a register which stores the page index to page table 26<sub>1</sub> or page table 26<sub>2</sub> mentioned later. The debugging register 15 is a register which is not used by OS20<sub>2</sub> or OS20<sub>1</sub> under execution and which is used as the register 18 for the data communications between OS's. In addition the registers 16 are a general register a flag register etc. The program counter 17 \*\*\*\*\*s the logical address which fetches a command from virtual memory space one time.

[0074]

In OS20<sub>1</sub> data check treating part 21<sub>1</sub> performs the check of whether to be inaccurate about the data which communicates from OS20<sub>1</sub> to OS20<sub>2</sub>. When this checked result is unusual alarm goes up and a series of processings are interrupted.

[0075]

Interruption distribution treating part 22<sub>1</sub> is made to jump to a predetermined interrupt handler with reference to interrupt vector table 25<sub>1</sub> based on the interruption number of the interrupt vector table register 13 at the time of interruption generating. Processing of interruption distribution treating part 22<sub>1</sub> is actually performed by control-section 11 inside at the time of interruption generating.

[0076]

Interrupt processing section 23<sub>1</sub> performs an interrupt handler and OS change processing (from OS20<sub>1</sub> to OS20<sub>2</sub>). Register data read-and-write treating part 24<sub>1</sub> OS<sub>1</sub> performs processing (processing which stores data) which writes the data which communicates to OS<sub>2</sub> in the debugging register 15 and processing (processing which acquires data) which reads the common data from OS<sub>2</sub> from the debugging register 15.

[0077]

Interrupt vector table 25<sub>1</sub> stores the logical address corresponding to the interruption processing for every interruption number mentioned above. Page table 26<sub>1</sub> is a table showing the correspondence relation between each virtual page of virtual-memory-space 40<sub>1</sub> (refer to drawing 3) and each physical page of the physical memory 12.

[0078]

Register save area 27<sub>1</sub> is a field for evacuating the contents of the register (the interrupt vector table register 13, the page table register 14, the other registers 16) corresponding to OS before a change at the time of OS change. When changing from OS20<sub>1</sub> to OS20<sub>2</sub>, specifically, the contents of the register corresponding to OS20<sub>1</sub> are evacuated to register save area 27<sub>1</sub>.

[0079]

On the other hand, in OS20<sub>2</sub>, data check treating part 21<sub>2</sub> performs the check of whether to be inaccurate about the data which communicates from OS20<sub>2</sub> to OS20<sub>1</sub>. When this checked result is unusual, an alarm goes up and a series of processings are interrupted.

[0080]

Interruption distribution treating part 22<sub>2</sub> is made to jump to a predetermined interrupt handler with reference to interrupt vector table 25<sub>2</sub> based on the interruption number of the interrupt vector table register 13 at the time of interruption generating. Processing of interruption distribution treating part 22<sub>2</sub> is actually performed by control-section 11 inside at the time of interruption generating.

[0081]

Interrupt processing section 23<sub>2</sub> performs an interrupt handler and OS change processing (from OS20<sub>2</sub> to OS20<sub>1</sub>). Register data read-and-write treating part 24<sub>2</sub>, OS<sub>2</sub> performs processing (processing which stores data) which writes the data which communicates to OS<sub>1</sub> in the debugging register 15, and processing (processing which acquires data) which reads the common data from OS<sub>1</sub> from the

debugging register 15.

[0082]

Interrupt vector table 25<sub>2</sub> stores the logical address corresponding to the interruption processing for every interruption number mentioned above. Page table 26<sub>2</sub> is a table showing the correspondence relation between each virtual page of virtual-memory-space 40<sub>2</sub> (refer to drawing 3) and each physical page of the physical memory 12.

[0083]

Register save area 27<sub>2</sub> is a field for evacuating the contents of the register (the interrupt vector table register 13, the page table register 14, the other registers 16) corresponding to OS before a change at the time of OS change. When changing from OS20<sub>1</sub> to OS20<sub>2</sub>, specifically, the contents of the register corresponding to OS20<sub>2</sub> are evacuated to register save area 27<sub>2</sub>.

[0084]

Below, with reference to drawing 5, the principle of operation of this Embodiment 1 is explained. In the example of the figure, OS20<sub>1</sub> shall be in an operating state and OS20<sub>2</sub> shall be in a waiting state. In this state, if an interrupt occurs in the OS20<sub>1</sub> side, based on the interruption number of the interrupt vector table register 13, interrupt vector table 25<sub>1</sub> will be referred to and an interrupt handler will be performed from L address of virtual-memory-space 40<sub>1</sub>.

[0085]

Henceforth, whenever it \*\*\*\*\*s the program counter 17 (refer to drawing 2), one time, address of virtual-memory-space 40<sub>1</sub> shifts and sequential execution of the command of the address concerned is carried out. And the data write command A is executed at Z address of virtual-memory-space 40<sub>1</sub> and the data which communicates to OS20<sub>2</sub> is written in the debugging register 15. Then the change command A is executed at N address and it changes from OS20<sub>1</sub> to OS20<sub>2</sub>.

[0086]

And if it will \*\*\*\*\* the program counter 17 one time and a logical address

will be the N+1st streetthe N+1st data reading commands B of virtual-memory-space 40<sub>2</sub> will be executedand data will be read from the debugging register 15. This data reading command B is good also as performingafter memorizing to addresses other than the N+1st and executing the N+1st commands.

[0087]

Henceforthwhenver it \*\*\*\*\*s the program counter 17 (refer to drawing 2) one time1 address of virtual-memory-space 40<sub>2</sub> shiftsand sequential execution of the command of the address concerned is carried out. And jump instruction is executed at S address of virtual-memory-space 40<sub>2</sub>and it is jumped to the N-M address of virtual-memory-space 40<sub>2</sub>.

[0088]

Henceforthwhenver it \*\*\*\*\*s the program counter 17 (refer to drawing 2) one time1 address of virtual-memory-space 40<sub>2</sub> shiftsand sequential execution of the command of the address concerned is carried out. And the data write command B is executed at P address of virtual-memory-space 40<sub>2</sub>and the data which communicates to OS20<sub>1</sub> is written in the debugging register 15. Then the change command B is executed at N addressand it changes from OS20<sub>2</sub> to OS20<sub>1</sub> (returned).

[0089]

And if it will \*\*\*\*\* the program counter 17 one time and a logical address will be the N+1st streetthe N+1st data reading commands A of virtual-memory-space 40<sub>1</sub> will be executedand data will be read from the debugging register 15. This data reading command A is good also as performingafter memorizing to addresses other than the N+1st and executing the N+1st commands.

[0090]

Then if it will \*\*\*\*\* the program counter 17 one time and a logical address will be the N+2nd street!RET will be performed and it will return from interruption. It operatesas well as \*\*\*\* when an interrupt occurs in the OS20<sub>2</sub> side.

[0091]

Belowwith reference to drawing 6 - drawing 8the example of this Embodiment 1

of operation is explained. Drawing 6 is a figure explaining the example of this Embodiment 1 of operation. In this example of operation the various commands corresponding to the processing shown in the figure are assigned to each logical address of virtual-memory-space 40<sub>1</sub> and virtual-memory-space 40<sub>2</sub>.

[0092]

In the example of the figure OS20<sub>1</sub> shall be in an operating state and OS20<sub>2</sub> shall be in a waiting state. When an interrupt occurs by the OS20<sub>1</sub> side in this state interruption distribution treating part 22<sub>1</sub> is 1. It comes out and jumps to the interruption processing concerned with reference to interrupt vector table 25<sub>1</sub> based on the interruption number of the interrupt vector table register 13.

[0093]

Thereby interrupt processing section 23<sub>1</sub> is 2. It comes out and an interrupt handler is performed from L address of virtual-memory-space 40<sub>1</sub>.

Henceforth whenever it \*\*\*\*\*s the program counter 17 (refer to drawing 2) one time 1 address of virtual-memory-space 40<sub>1</sub> shifts and sequential execution of the command of the address concerned is carried out.

[0094]

And in Y address of virtual-memory-space 40<sub>1</sub> interrupt inhibit instruction is executed before OS change processing in virtual-memory-space 40<sub>1</sub>. If interrupt inhibit instruction is executed it will be in the state where other interruption is not received. While performing OS change processing this interrupt inhibit instruction is executed in order to prevent interruption with a higher priority from entering multiply.

[0095]

And in OS change processing (a register is evacuated to a register save area) the contents of the interrupt vector table register 13 corresponding to OS20<sub>1</sub> the page table register 14 and the other registers 16 are evacuated to register save area 27<sub>1</sub> at the Y+1st street.

[0096]

In OS spawn process (OS's communication data check) in the Z-1st continuing

streetthe unjust check etc. of the data which communicates from OS20<sub>1</sub> to OS20<sub>2</sub> are performed. When a checked result is unusualalarm goes up and a series of processings are interrupted.

[0097]

In this casea checked result considers it as a normal thingand writes in the data which communicates from OS20<sub>1</sub> to OS20<sub>2</sub> to the debugging register 15 in OS spawn process (OS's communication data check) of Z address.

[0098]

And in OS change processing (IDTR change command)the interrupt vector table register 13 is changed from the object for OS20<sub>1</sub> to OS20<sub>2</sub>. In the next OS change processing (page table register variation order)the page table register 14 is changed into OS20<sub>2</sub> from the object for OS20<sub>1</sub> at N address of virtual-memory-space 40<sub>1</sub>. Therebyit changes from OS20<sub>1</sub> to OS20<sub>2</sub> (3). .

[0099]

In continuing OS change processing (the commo data between OS's is read from a register)data is read from the debugging register 15 at the N+1st street of virtual-memory-space 40<sub>2</sub>.

[0100]

In subsequent OS change processing (a register is returned from a register save area). the contents of the register for OS20<sub>2</sub> evacuated to register save area 27<sub>2</sub> in the N+2nd street before -- the interrupt vector table register 13 and the page table register 14 -- and -- in addition to this -- it returns to the register 16respectively. And an interruption permission command is executed in the N+3rd street. If an interruption permission command is executedit will be in the state of receiving other interruption (4). .

[0101]

And interrupt processing section 23<sub>2</sub> carries out sequential execution of the interrupt handler. And jump instruction is executed at S address of virtual-memory-space 40<sub>2</sub>and it is jumped to the N-M address of virtual-memory-space 40<sub>2</sub> (5). .

[0102]

Then interrupt inhibit instruction in the N-M address of virtual-memory-space 40<sub>2</sub> is executed and it will be in the state where other interruption is not received. And in OS change processing (a register is evacuated to a register save area) of 1st [ + ] N-M the contents of the interrupt vector table register 13 corresponding to OS20<sub>2</sub> the page table register 14 and the other registers 16 are evacuated to register save area 27<sub>2</sub>.

[0103]

In the P-1st continuing OS change processings (OS's communication data check) the unjust check etc. of the data which communicates from OS20<sub>2</sub> to OS20<sub>1</sub> are performed. When a checked result is unusual alarm goes up and a series of processings are interrupted.

[0104]

In this case a checked result considers it as a normal thing and writes in the data which communicates from OS20<sub>2</sub> to OS20<sub>1</sub> to the debugging register 15 in OS spawn process (the commo data between OS's is written in a register) of P address. And in OS change processing (IDTR change command) the interrupt vector table register 13 is changed from the object for OS20<sub>2</sub> to OS20<sub>1</sub> (6). .

[0105]

In continuing OS change processing (page table register variation order) the page table register 14 is changed into OS20<sub>1</sub> from the object for OS20<sub>2</sub> at N address of virtual-memory-space 40<sub>2</sub>. Thereby it changes from OS20<sub>2</sub> to OS20<sub>1</sub> (7). .

[0106]

In subsequent OS change processing (the commo data between OS's is read from a register) data is read from the debugging register 15 in the N+1st street of virtual-memory-space 40<sub>1</sub>.

[0107]

And in OS change processing (a register is returned from a register save area). the contents of the register for OS20<sub>1</sub> previously evacuated to register save area 27<sub>1</sub> in the N+2nd street of virtual-memory-space 40<sub>1</sub> – the interrupt vector table

register 13 and the page table register 14 -- and -- in addition to this -- it returns to the register 16 respectively. And an interruption permission command is executed in the N+3rd street. If an interruption permission command is executed it will be in the state of receiving other interruption.

[0108]

Next interrupt processing section 23<sub>1</sub> carries out sequential execution of the interrupt handler. And IRET (interruption return instruction) is performed at X address of virtual-memory-space 40<sub>1</sub> and a manipulation routine usually returns to a manipulation routine from an interrupt handler (8). .

[0109]

Drawing 7 and drawing 8 are the flow charts explaining the above-mentioned example of operation. In the figure if an interrupt occurs an application run of OS20<sub>1</sub> will be stopped by step SA1 and interruption distribution treating part 22<sub>1</sub> will be started by it. In step SA2 interruption distribution treating part 22<sub>1</sub> of OS20<sub>1</sub> acquires the interruption number corresponding to interruption from the interrupt vector table register 13.

[0110]

In step SA3 interrupt vector table 25<sub>1</sub> is referred to by interruption distribution treating part 22<sub>1</sub> and it is jumped by the logical address with the interrupt handler in virtual-memory-space 40<sub>1</sub> (1 of drawing 6). . An interrupt handler is performed in step SA4 (2 of drawing 6). .

[0111]

Interrupt inhibit instruction is executed in step SA5 (Y address of drawing 6). In step SA6 OS change processing (a register is evacuated to a register save area) is performed. In step SA7 OS change processing (OS's communication data check) is performed and the check of the data which communicates to OS20<sub>2</sub> is performed (the Z-1st street of drawing 6). When a checked result is unusual alarm goes up and a series of processings are interrupted.

[0112]

In this case a checked result considers it as a normal thing in step SA8 OS change

processing (the commo data between OS's is written in a register) is performed and the data which communicates to OS20<sub>2</sub> to the debugging register 15 is written in (Z address of drawing 6).

[0113]

In step SA9OS change processing (IDTR change command) is performed. In step SA10OS change processing (page table register variation order) is performed (3 of drawing 6). It changes from OS20<sub>1</sub> to OS20<sub>2</sub>.

[0114]

OS change processing (the commo data between OS's is read from a register) is performed by OS20<sub>2</sub> step SA11 (the N+1st street of drawing 6). In step SA12OS change processing (a register is returned from a register save area) is performed (the N+2nd street of drawing 6).

[0115]

An interruption permission command is executed in step SA13 (the N+3rd street of drawing 6). An interrupt handler is performed in step SA14. In step SA15jump instruction is executed and it is jumped by the N-M address (5 of drawing 6). .

[0116]

Interrupt inhibit instruction is executed in step SA16 shown in drawing 8 (N-M address of drawing 6). In step SA17OS change processing (a register is evacuated to a register save area) is performed. In step SA18OS change processing (OS's communication data check) is performed and the check of the data which communicates to OS20<sub>1</sub> is performed (the P-1st street of drawing 6). When a checked result is unusual alarm goes up and a series of processings are interrupted.

[0117]

In this case a checked result considers it as a normal thing in step SA19OS change processing (the commo data between OS's is written in a register) is performed and the data which communicates to OS20<sub>2</sub> to the debugging register 15 is written in (P address of drawing 6).

[0118]

In step SA20OS change processing (IDTR change command) is performed. In step SA21OS change processing (page table register variation order) is performed (7 of drawing 6). It changes from OS20<sub>2</sub> to OS20<sub>1</sub>.

[0119]

OS change processing (the common data between OS's is read from a register) is performed by OS20<sub>1</sub> step SA22 (the N+1st street of drawing 6). In step SA23OS change processing (a register is returned from a register save area) is performed (the N+2nd street of drawing 6).

[0120]

An interruption permission command is executed in step SA24 (the N+3rd street of drawing 6). An interrupt handler is performed in step SA25. In step SA26IRET (interruption return instruction) is performed and it returns from interruption.

[0121]

As explained above according to this Embodiment 1 it is from OS20<sub>1</sub> to OS20<sub>2</sub> (.). Or the data which communicates from OS20<sub>2</sub> to OS20<sub>1</sub> is stored in the debugging register 15 and it is from OS20<sub>1</sub> to OS20<sub>2</sub> (.). Or when a change is performed from OS20<sub>2</sub> to OS20<sub>1</sub> it is OS20<sub>2</sub> (.). Or since OS20<sub>1</sub> reads data from the debugging register 15 since the data size stored in the debugging register 15 is restricted it is hard to spread inaccurate data and a virus program between OS20<sub>1</sub> and OS20<sub>2</sub> and the security of a multi operating system and reliability can be raised.

[0122]

According to this Embodiment 1 the normality of the data of a communication object is checked by data check treating part 21<sub>1</sub> and data check treating part 21<sub>2</sub>. Since it is used making communications processing continue only when normal the security of a multi operating system and reliability can be raised further.

[0123]

Since the register which is not used by OS is only used according to this Embodiment 1 a special device or circuit are not needed at all but a commercial personal computer can also realize the preparation.

[0124]

(Embodiment 2)

Now although Embodiment 1 mentioned above showed the case where data was communicated via a register between OS20<sub>1</sub> and OS20<sub>2</sub>. The data communications which used the register when having communicated data and data size was size storable in a register are performed and when it is size unstorable in a register it may constitute so that the data may be communicated with an option.

[0125]

Below when it is data size unstorable in a register the example of composition which communicates by changing from communication by a register to communication using NIC (Network Interface Card) is explained.

[0126]

Drawing 9 is a block diagram showing the composition of this Embodiment 2. The multi operating system shown in this figure NIC (Network Interface Card) 60 NIC device driver 70<sub>1</sub> It has the hardware 10 shown in NIC device driver 70<sub>2</sub> OS20<sub>1</sub> OS20<sub>2</sub> AP30<sub>1</sub> - 30<sub>4</sub> the network 50 and drawing 2 (a graphic display is omitted).

[0127]

Here OS20<sub>1</sub> and OS20<sub>2</sub> are the same as that of OS20<sub>1</sub> and OS20<sub>2</sub> which were shown in drawing 2 and have data size judgment part 29<sub>1</sub> and data size judgment part 29<sub>2</sub> further respectively.

[0128]

OS change treating part 28<sub>1</sub> It corresponds to data check treating part 21<sub>1</sub> shown in drawing 2 interrupt distribution treating part 22<sub>1</sub> interrupt processing section 23<sub>1</sub> interrupt vector table 25<sub>1</sub> page table 26<sub>1</sub> register save area 27<sub>1</sub> etc. Data check treating part 21<sub>2</sub> which showed drawing 2 OS change treating part 28<sub>2</sub> It corresponds to interrupt distribution treating part 22<sub>2</sub> interrupt processing section 23<sub>2</sub> interrupt vector table 25<sub>2</sub> page table 26<sub>2</sub> register save area 27<sub>2</sub> etc.

[0129]

In the multi operating system the change to OS2<sub>2</sub> from OS2<sub>1</sub> or OS2<sub>1</sub> from OS2<sub>2</sub> is performed in an interrupt handler. As a generation factor of interruption the needed information of the data between OS's the periodical change demand by a timer (graphic display abbreviation) etc. are mentioned.

[0130]

In this multi operating system like before a common control program The feature is at the point of realizing delivery of data between OS2<sub>1</sub> and OS2<sub>2</sub> where high security is held using register 18 for data communications between OS's or NIC60 without passing a shared memory etc.

[0131]

Hereafter each component is explained in full detail. NIC60 is a LAN (Local Area Network) card and is provided with the communication-interface function for example. This NIC60 is controlled by NIC device driver 70<sub>1</sub> and NIC device driver 70<sub>2</sub> which are mentioned later.

[0132]

The NIC transmission processing part 61 is provided with the function which transmits a packet in NIC60. The transmission buffer 62 stores temporarily the packet transmitted from the NIC transmission processing part 61. The NIC receiving processing part 63 is provided with the function to receive a packet from the network 50. The receive buffer 64 stores temporarily the packet received by the NIC receiving processing part 63.

[0133]

The transmitting buffer register 65 is a register which stores the value (address) corresponding to either transmission buffer 72<sub>1</sub> mentioned later or transmission buffer 72<sub>2</sub>. The receiving buffer register 66 is a register which stores the value (address) corresponding to either receive buffer 74<sub>1</sub> mentioned later or receive buffer 74<sub>2</sub>.

[0134]

NIC device driver 70<sub>1</sub> realizes the function of a part of OS2<sub>1</sub> and is provided

with the function which controls NIC60 at the time of communicationthe function which rewrites the value of the transmitting buffer register 65 or the receiving buffer register 66etc.

[0135]

Transmission processing part 71<sub>1</sub> is provided with the function which transmits the data from OS20<sub>1</sub> in NIC device driver 70<sub>1</sub>. Transmission buffer 72<sub>1</sub> stores the data from OS20<sub>1</sub> temporarily. Receiving processing part 73<sub>1</sub> is provided with the function to receive the data addressed to OS20<sub>1</sub>. Receive buffer 74<sub>1</sub> stores the data addressed to OS20<sub>1</sub> temporarily.

[0136]

On the other handNIC device driver 70<sub>2</sub> realizes the function of a part of OS20<sub>2</sub> and is provided with the function which controls NIC60 at the time of communicationthe function which rewrites the value of the transmitting buffer register 65 or the receiving buffer register 66etc.

[0137]

Here192.168.1.3 is given to OS20<sub>1</sub> as an IP addressfor example. On the other hand192.168.1.4 is given to OS20<sub>2</sub> as another IP addressfor example.

[0138]

Belowthe principle of operation in the case of communicating data using NIC60 is explained with reference to the block diagram shown in drawing 10. Herethe case where it communicates from OS20<sub>1</sub> to OS20<sub>2</sub> is explained via NIC60.

[0139]

It shall be in the state where OS20<sub>1</sub> is performingand the value corresponding to transmission buffer 72<sub>1</sub> is stored in the transmitting buffer register 66and the value corresponding to receive buffer 74<sub>1</sub> is stored in the receiving buffer register 65in the figure.

[0140]

In this statetransmission processing part 71<sub>1</sub> of NIC device driver 70<sub>1</sub> stores the data from OS20<sub>1</sub> in transmission buffer 72<sub>1</sub> (1). This data is transmitted to OS20<sub>2</sub> from OS20<sub>1</sub>. Thereforethe transmission destination IP address of the data

concerned is 192.168.1.4 given to OS20<sub>2</sub>.

[0141]

In (2)transmission processing part 71<sub>1</sub> rewrites the value of the receiving buffer register 65 of NIC60 from receive buffer 74<sub>1</sub> to receive buffer 74<sub>2</sub>. In (3)the NIC transmission processing part 61 copies data to the transmission buffer 62 from transmission buffer 72<sub>1</sub>after accessing transmission buffer 72<sub>1</sub> with reference to the transmitting buffer register 66.

[0142]

In (4)the NIC transmission processing part 61 reads data from the transmission buffer 62and transmits this to the network 50. And the data concerned is stored in the receive buffer 64 after being received by the NIC receiving processing part 63.

[0143]

In (5)the NIC receiving processing part 63 copies data to receive buffer 74<sub>2</sub> from the receive buffer 64after accessing receive buffer 74<sub>2</sub> with reference to the receiving buffer register 65.

[0144]

In (6)OS spawn process part 28<sub>1</sub> changes OS from OS20<sub>1</sub> to OS20<sub>2</sub>. In (7)receiving processing part 73<sub>2</sub> reads data from receive buffer 74<sub>2</sub>and passes it to OS20<sub>2</sub> after changing this data.

[0145]

Thuswithout passing a common control programa shared memoryetc. between OS's like beforeoperation of communication through NIC60 can realize communication between OS'swhere high security is held.

[0146]

Belowthe example of the data communications between OS's of this Embodiment 2 of operation is explained. Drawing 11 and drawing 12 are flow charts which show the example of the data communications between OS's of this Embodiment 2 of operation. In this example of operationcommunication by register 18 for data communications between OS's or NIC60 is changed with the size of dataand is

performed. Below the case where the debugging register 15 is used as the register 18 for the data communications between OS's is explained like Embodiment 1.

[0147]

In drawing 11 if an interrupt occurs an application run of OS20<sub>1</sub> will be stopped by step SB1 and interruption distribution treating part 22<sub>1</sub> will be started by it. In step SB2 interruption distribution treating part 22<sub>1</sub> of OS20<sub>1</sub> acquires the interruption number corresponding to interruption from the interrupt vector table register 13.

[0148]

In step SB3 interrupt vector table 25<sub>1</sub> is referred to by interruption distribution treating part 22<sub>1</sub> and it is jumped by the logical address with the interrupt handler in virtual-memory-space 40<sub>1</sub>. An interrupt handler is performed in step SB4.

[0149]

Interrupt inhibit instruction is executed in step SB5. In step SB6 OS change processing (a register is evacuated to a register save area) is performed. In step SB7 OS change processing (OS's communication data check) is performed and the check of the data which communicates to OS20<sub>2</sub> is performed. When a checked result is unusual alarm goes up and a series of processings are interrupted.

[0150]

In this case a checked result considers it as a normal thing OS change processing (data size check of the commo data between OS's) is performed in step SB8 and it is investigated whether the data which communicates to OS<sub>2</sub> to the debugging register 15 can be written in.

[0151]

When the decision result of step SB9 can write in by "Yes" by step SB10 OS change processing (the commo data between OS's is written in a register) is performed and the data which communicates to OS20<sub>2</sub> to the debugging register 15 is written in. In step SB12 OS change processing (IDTR change command) is performed.

[0152]

When the decision result of step SB9 cannot write in by "No" OS change processing (commo data transmitting processing using NIC60) is performed by step SB11. In this processing processings from (1) to (5) shown in drawing 10 are performed and it shifts to step SB12 after that.

[0153]

In step SB13 OS change processing (page table register variation order) is performed and it changes from OS20<sub>1</sub> to OS20<sub>2</sub>. In step SB14 it is judged whether it is the communication which used the debugging register 15.

[0154]

For example in communicating using NIC60a decision flag is transmitted with commo data and OS20<sub>2</sub> receives the decision flag and it detects communication by NIC60. When a decision flag is not received the judgment which used the debugging register 15 that it is communication is performed. Not only this but other judgment methods may be used.

[0155]

When the decision result of step SB14 is "Yes" OS change processing (the commo data between OS's is read from a register) is performed by OS20<sub>2</sub> step SB15. In step SB17 OS change processing (a register is returned from a register save area) is performed.

[0156]

When the decision result of step SB14 is "No" OS change processing (commo data reception using NIC60) is performed by step SB16. In this processing processings of (7) shown in drawing 10 is performed and it shifts to step SB17 after that.

[0157]

An interruption permission command is executed in step SB18. An interrupt handler is performed in step SB19. In step SB20 jump instruction is executed and it is jumped by the N-M address.

[0158]

Interrupt inhibit instruction is executed in step SB21 shown in drawing 12. In step SB22 OS change processing (a register is evacuated to a register save area) is performed. In step SB23 OS change processing (OS's communication data check) is performed and the data which communicates to OS20<sub>1</sub> is checked. When a checked result is unusual alarm goes up and a series of processings are interrupted.

[0159]

In this case a checked result considers it as a normal thing OS change processing (data size check of the commo data between OS's) is performed in step SB24 and it is investigated whether the data which communicates to OS<sub>1</sub> to the debugging register 15 can be written in.

[0160]

When the decision result of step SB25 can write in by "Yes" by step SB26 OS change processing (the commo data between OS's is written in a register) is performed and the data which communicates to OS20<sub>1</sub> to the debugging register 15 is written in. In step SB28 OS change processing (IDTR change command) is performed.

[0161]

When the decision result of step SB25 cannot write in by "No" OS change processing (commo data transmitting processing using NIC60) is performed by step SB27. In this processing by the same method as processings from (1) to (5) shown in drawing 10 transmitting processing of the commo data from OS20<sub>2</sub> to OS20<sub>1</sub> is performed and it shifts to step SB28 after that.

[0162]

In step SB29 OS change processing (page table register variation order) is performed and it changes from OS20<sub>2</sub> to OS20<sub>1</sub>. In step SB30 it is judged whether it is the communication which used the debugging register 15.

[0163]

For example in communicating using NIC60a decision flag is transmitted with commo data and OS20<sub>1</sub> receives the decision flag and it detects communication

by NIC60. When a decision flag is not received the judgment which used the debugging register 15 that it is communication is performed. Not only this but other judgment methods may be used.

[0164]

When the decision result of step SB30 is "Yes" OS change processing (the commo data between OS's is read from a register) is performed by OS20<sub>1</sub> step SB31. In step SB33 OS change processing (a register is returned from a register save area) is performed.

[0165]

When the decision result of step SB30 is "No" OS change processing (commo data reception using NIC60) is performed by step SB32. In this processing by the same method as processing of (7) shown in drawing 10 OS20<sub>1</sub> performs reception of commo data and shifts to step SB33 after that.

[0166]

An interruption permission command is executed in step SB34. An interrupt handler is performed in step SB35. In step SB36 IRET (interruption return instruction) is performed and it returns from interruption.

[0167]

As explained above when according to this Embodiment 2 communicating data and data size is size storable in the debugging register 15 In being the size which performs the data communications using the debugging register 15 and cannot be stored in the debugging register 15 Since the data is communicated using NIC60 like before A common control program Without passing a shared memory etc. using register 18 for data communications between OS's or NIC60 where high security is held data can be delivered between OS20<sub>1</sub> and OS20<sub>2</sub>.

[0168]

Since according to this Embodiment 2 the normality of the data of a communication object was checked and it used making communications processing continue only when normal the security of a multi operating system and reliability can be raised further.

[0169]

Data size was large by this Embodiment 2 when it was not able to store in the debugging register 15 decided to communicate by using NIC 60 here but. It is good also as not being limited to it but performing data communications via a common control program shared memory etc. like before.

[0170]

In the above in performing data communications via a common control program shared memory etc. Although security will fall data size is small and when communicating using the debugging register 15 where still high security is held data can be delivered between OS 2<sub>1</sub> and OS 2<sub>2</sub>.

[0171]

Drawing 13 (a) - drawing 13 (d) are the figures explaining the applications 1-4 of Embodiment 1 or 2 mentioned above. The application 1 shown in drawing 13 (a) is an example which old and new OS is made to live together and is used at the time of system renewal. New OS supports OS 2<sub>1</sub> for example. On the other hand the old OS supports OS 2<sub>1</sub>.

[0172]

The application 2 shown in drawing 13 (b) is an example which carries out speed development of the new function by the sauce public presentation OS side without utilizing sauce public presentation OS and waiting for the release of sauce secret OS. Sauce secret OS is an operating system with which the source code is not exhibited for example supports OS 2<sub>1</sub>. On the other hand sauce public presentation OS is an operating system with which the source code is exhibited and supports OS 2<sub>2</sub>.

[0173]

The application 3 shown in drawing 13 (c) is an example which makes exclusive OS and a versatile OS live together. Exclusive OS shares the function to specialize in real time nature etc. for example supports OS 2<sub>1</sub>. On the other hand a versatile OS is Windows (registered trademark) of Microsoft Corp. etc. shares a GUI function and supports OS 2<sub>2</sub>.

[0174]

The application 4 shown in drawing 13 (d) is an example which divides resources and is an example from which directions for use differ by the purpose of using OS1 and OS2. OS1 supports OS20<sub>1</sub> for example. On the other hand OS2 supports OS20<sub>2</sub>.

[0175]

Drawing 14 is a figure explaining the case where Embodiment 1 or 2 is applied to system shift. It is necessary to make all the modules of the old OS mounted in the old terminal shift to new OS of a new terminal at once and anxiety remains by a safety aspect by the conventional shift in the figure.

[0176]

On the other hand each module of the old OS mounted in the old terminal is made to shift if there is a multi operating system checking safety to the old OS of a way station. Next each module of the old OS is made to shift in a way station checking safety to new OS.

[0177]

Next all the modules of new OS of a way station are made to shift to new OS of a new terminal at once. In this case since it is the shift between new OS's a problem does not occur. The shift by multi-OS is effective when it gives a new function to a certain specific module in a hurry.

[0178]

Drawing 15 is a figure explaining the case where Embodiment 1 or 2 is applied to a high security gateway. In the figure hardware is provided with the multi operating system (OS1 and OS2) and is connected to the Internet via NIC (Network Interface Card). This hardware functions as a high security gateway.

[0179]

OS1 accumulates a communication log in DK(disk) 1 while managing AP(application program) 1 which a user uses. On the other hand OS2 supervises the communication packet from the Internet and it accumulates a surveillance log in DK2.

[0180]

In OS2 it communicates by controlling NIC directly. For this reason OS2 provides OS1 with false NIC-I/F which carried out false [ of the offer interface for software of NIC ] in order to support communication of OS1. AP2 is an application program for packet monitoring log collection and it operates on OS2.

[0181]

AP2 only looks at a communication packet from outside and it is not infected by a virus in order not to process the execution code contained in a communication packet. The alteration by a holder in bad faith was not made and the surveillance log accumulated in DK2 is come. Therefore it becomes possible to leave an offensive trace.

[0182]

Here when an attack of security is delivered to a user's environment (OS1 grade) it is possible that the communication log of OS1 is altered. However in Embodiment 1 or 2 since the surveillance log of OS1 and OS2 is managed independently a multi operating system enables it to pursue an aggressor. Thereby the effect which deters the attack to the security gateway concerned is expectable.

[0183]

Drawing 16 is a figure explaining the case where Embodiment 1 or 2 is applied to a desktop Grid terminal. In the figure hardware is provided with the multi operating system (OS1 and OS2) and is connected to the Internet via NIC. The Grid server is connected to the Internet. Hardware functions as a desktop Grid terminal.

[0184]

A desktop Grid terminal is a terminal for performing desktop Grid calculation which is going to assign a part of big calculation and is going to realize calculation while a user does not use a computer. Here if it sees from the user side the above-mentioned calculation the contents of whose are unknown will become anxious [ the adverse effect which it has on own computer environment ].

[0185]

So in the figure the above-mentioned adverse effect can be eliminated by managing own computer environment by OS1 and managing the processing environment of Grid calculation by OS2. That is all of data and the program which are used by Grid calculation are managed and performed only by the controller of OS2. On the other hand access is not permitted to a computer resource called DK1 and MEM(memory) 1 under control of OS1. Therefore the adverse effect from the program of Grid calculation is eliminated.

[0186]

Drawing 17 is a figure explaining the case where Embodiment 1 or 2 is applied to a remote management terminal. In the figure hardware is provided with the multi operating system (OS1 and OS2) and is connected to the network via NIC. A system administrator machine and other machines are connected to the network. Hardware functions as a remote management terminal.

[0187]

In the figure the category which a user can manage is restricted to OS1 and it is constituted so that a system administrator machine may perform management of OS2 via a network. Thereby a system administrator can build the default environment of the personal computer used in office environment etc. under management of OS2 and on the other hand a user can build the environment used according to a user individual's liking and situation under management of OS1.

[0188]

Therefore the environment prepared by the system side is prevented from becoming fault of operation by a user individual's configuration. OS2 provides the environment which was managed by the system administrator by the system administrator machine and was designed by the system side. When using this environment the reads the file information of OS2 into OS1 and a user starts it or requests starting from OS2 and performs with the gestalt of a client/server.

[0189]

Drawing 18 is a figure explaining the case where Embodiment 1 or 2 is applied to an efficient net service provision terminal. In the figure hardware is provided with

the multi operating system (OS1 and OS2) and is connected to the network via NIC. A contents provider server and other machines are connected to the network. Hardware functions as an efficient net service provision terminal.

[0190]

In the figure an efficient net service provision terminal. Like a remote management terminal (refer to drawing 17) one OS1 as the bottom of management of net service of OS2 of another side under a user's management. For example, the contents before a release are distributed under management of OS2 (DK2) a priori and it is a terminal which realizes instant use of the contents from DK1 under management of OS1 at released time.

[0191]

After contents are distributed to DK2 under management of OS2 from a contents provider server via a network, the contents concerned are provided by a user's hope DK1 under management of DK2 to OS1 and use of them is attained instantly. Thus, in a contents provider server, concentration of download access is prevented by advance distribution.

[0192]

Drawing 19 is a figure explaining the case where Embodiment 1 or 2 is applied to a high security Web Service offer server. In the figure, hardware is provided with the multi operating system (OS1 and OS2) and is connected to the network via NIC. The Web reading terminal is connected to the network. Hardware functions as a high security Web Service offer server.

[0193]

In the high security Web Service offer server of the figure, the environment which can be accessed from the outside is OS2 and the environment which can access only a local user is built by OS1.

[0194]

For example, Web open content and a utilization log are stored in DK2 under management of OS2. On the other hand, data not to exhibit is stored in DK1 under management of OS1. In this case, the situation of releasing information not to

release on Web accidentally by a setting error etc. is avoided. Even if it is a case where OS2 side receives the attack by a holder in bad faith from the exterior in OS1 it becomes possible to operate except for a network function.

[0195]

Although Embodiments 1 and 2 which start this invention above have been explained in full detail with reference to drawings the concrete example of composition is not restricted to these Embodiments 1 and 2 and even if the design variation etc. of the range which does not deviate from the gist of this invention occurs it is included in this invention.

[0196]

For example in Embodiments 1 and 2 mentioned above The program for realizing multi operating system control control etc. which were mentioned above is recorded on the recording medium 200 which was shown in drawing 20 and in which computer reading is possible The program recorded on this recording medium 200 may be made to read into the computer 100 shown in the figure and each function may be realized by performing.

[0197]

CPU 110 in which the computer 100 shown in the figure executes the above-mentioned program The input devices 120 such as a keyboard and a mouse and ROM (Read Only Memory) 130 which memorize various data all comprises the bus 170 which connects each part of a device with RAM (Random Access Memory) 140 which memorizes computation parameters etc. the reader 150 which reads a program in the recording medium 200 and the output units 160 such as a display and a printer.

[0198]

CPU 110 realizes the function mentioned above by executing a program after reading the program currently recorded on the recording medium 200 via the reader 150. As the recording medium 200 an optical disc a flexible disc a hard disk etc. are mentioned.

[0199]

In Embodiments 1 and 2 as how to change between two operating systems Although the logical address of virtual memory space was set up appropriately and how to change an operating system by changing an interrupt vector table register and a page table register was explained This invention is not restricted to this and may change an operating system with a conventional virtual-machine method or microkernel method.

[0200]

In Embodiments 1 and 2 although the example of composition which has two OS<sub>20 1</sub> shown in drawing 1 and OS<sub>20 2</sub> as a multi operating system was explained the example of composition which has three or more OS's is also included in this invention.

[0201]

For example in composition of having OS<sub>1</sub> OS<sub>2</sub> OS<sub>3</sub>...n OS's of OS<sub>n</sub>. When it interrupts during OS<sub>1</sub> run and (1) occurs change to OS<sub>2</sub> from OS<sub>1</sub> when it interrupts during OS<sub>1</sub> run and (2) occurs change to OS<sub>3</sub> from OS<sub>1</sub> and it is made to be the same as that of the following What is necessary is just to control to change from OS<sub>1</sub> to OS<sub>n</sub> when an interrupt (n-1) occurs during OS<sub>1</sub> run.

[0202]

[Effect of the Invention]

As explained above according to this invention the data which becomes a predetermined register with a communication object is stored during execution of the 1st operating system Since we decided to acquire the data which serves as a communication object from a predetermined register when a change was made by the 2nd operating system from the 1st operating system By using for communication between operating systems the register in which the size of storable data was restricted without passing a shared memory. The possibility of the abnormal operation of each operating system by communication of inaccurate data can be avoided and the effect that the security of a multi operating system and reliability can be raised is done so.

[0203]

When the data used as a communication object is stored in a predetermined register according to this invention. Virtual memory was beforehand carried out to logical address N of the 1st virtual memory space corresponding to the 1st operating system. With the 1st change command changed from the 1st operating system to the 2nd operating system. It changes from the 1st operating system to the 2nd operating system. The data acquisition command memorized beforehand in the logical address N+1 of the 2nd virtual memory space corresponding to the 2nd operating system. Or since the data acquisition command executed after the command memorized beforehand in this logical address N+1 is executed and the data which serves as a communication object from a predetermined register is acquired. Intersectionssuch as the conventional base OS and VM monitorare unnecessarythe 1st and 2nd operating systems live togetherand the effect that the security of a multi operating system and reliability can be raised is done so. [0204]

According to this inventionthe 2nd operating systemWith the 2nd change command for changing from the 2nd operating system by which virtual memory was carried out beforehand to the 1st operating systemafter executing a data acquisition command. The effect that a change and cut return of an operating system can be performed smoothly is done somaintaining high security and reliabilitysince it changes from the 2nd operating system to the 1st operating system.

[0205]

According to this inventionstore the data used as a communication object in the register which backs up the data stored in the predetermined register and by which data was backed upand with the 2nd change command. Since we decided to restore the backed-up data to the predetermined register in which data was stored when a change was performed to the 1st operating system from the 2nd operating systemBy being able to store the data which communicates between operating systems in the register currently used during execution of the 1st operating systemand communicating using the registerThe effect that the

security of a multi operating system and reliability can be raised is done so.

[0206]

Since it presupposed that it is a predetermined register a register which is not used by the 1st operating system during execution of the 1st operating system according to this inventionThe data which communicates between operating systems can be stored and the effect that the security of a multi operating system and reliability can be raised is done so by communicating using the register.

[0207]

Since it presupposed that it is a predetermined register the debugging register and/or general register which are not used by the 1st operating system during execution of the 1st operating system according to this inventionBy being able to store in a debugging register and/or a general register the data which communicates between operating systemsand communicating using the registerThe effect that the security of a multi operating system and reliability can be raised is done so.

[0208]

According to this inventionthe size of the data used as a communication object judges whether it is size storable in a predetermined registerWhen it can storestore data in a predetermined registerand when it cannot storeSince the data which serves as a communication object by storing means other than a predetermined register is stored and the 2nd operating system acquires the stored dataA suitable storing means can be chosen according to the size of dataand the effect that the data between the efficient operating systems which have high security and reliability can be communicated is done so.

[0209]

When the data used as said communication object addressed to the 2nd operating system is stored in the storing means with which the Network Interface Card was equipped according to this inventionSince the 1st operating system is relayed and connection of a storing means is set as the 2nd operating systemEven if the size of the data used as a communication object is size

unstorable in a predetermined register the effect that the data between the efficient operating systems which have high security and reliability can be communicated is done so.

[0210]

Since it was made to make a computer perform the multi operating system control method that even either of the above-mentioned inventions was indicated according to this invention Computer reading of a program becomes possible and it does so the effect that any one operation of the above-mentioned invention by this can be performed by computer.

[Brief Description of the Drawings]

[Drawing 1] It is a block diagram showing the outline composition of Embodiment 1 concerning this invention.

[Drawing 2] It is a block diagram showing the concrete composition of Embodiment 1.

[Drawing 3] It is a figure explaining memory mapping between the multi operating systems in Embodiment 1.

[Drawing 4] It is a figure explaining the principle of operation of Embodiment 1.

[Drawing 5] It is a figure explaining the principle of operation of Embodiment 1.

[Drawing 6] It is a figure explaining the example of Embodiment 1 of operation.

[Drawing 7] It is a flow chart explaining the example of Embodiment 1 of operation.

[Drawing 8] It is a flow chart explaining the example of Embodiment 1 of operation.

[Drawing 9] It is a block diagram showing the composition of Embodiment 2 concerning this invention.

[Drawing 10] It is a block diagram explaining the example of Embodiment 2 of operation.

[Drawing 11] It is a flow chart explaining the example of Embodiment 2 of operation.

[Drawing 12] It is a flow chart explaining the example of Embodiment 2 of operation.

[Drawing 13] It is a figure explaining the applications 1-4 of Embodiment 1 or 2

concerning this invention.

[Drawing 14] It is a figure explaining the case where Embodiment 1 or 2 is applied to system shift.

[Drawing 15] It is a figure explaining the case where Embodiment 1 or 2 is applied to a high security gateway.

[Drawing 16] It is a figure explaining the case where Embodiment 1 or 2 is applied to a desktop Grid terminal.

[Drawing 17] It is a figure explaining the case where Embodiment 1 or 2 is applied to a remote management terminal.

[Drawing 18] It is a figure explaining the case where Embodiment 1 or 2 is applied to an efficient net service provision terminal.

[Drawing 19] It is a figure explaining the case where Embodiment 1 or 2 is applied to a high security Web Service offer server.

[Drawing 20] It is a block diagram showing the composition of the modification of Embodiment 1 or 2.

[Drawing 21] It is a block diagram showing the example 1 of composition of the conventional multi operating system.

[Drawing 22] It is a block diagram showing the example 2 of composition of the conventional multi operating system.

[Description of Notations]

10 Hardware

11 Control section

12 Physical memory

12a<sub>1</sub> the physical memory field for 12a<sub>2</sub>OS

12b<sub>1</sub> the physical memory field for 12b<sub>2</sub>AP

13 Interrupt vector table register

14 Page table register

15 Debugging register

16 Other registers

17 Program counter

18 The register for the data communications between OS's

20 120 2 OS

21 121 2 data check treating part

22 122 2 interruption distribution processing part

23 123 2 interrupt processing section

24 124 2 register data read-and-write treating part

25 125 2 interrupt vector table

26 126 2 page table

27 127 2 register save area

28 1a 28 2 OS change treating part

29 129 2 data size judgment part

30 130 2 AP

40 140 2 virtual memory space

40a 1 virtual memory space for 40a 2 OS

40b 1 virtual memory space for 40b 2 AP

50 Network

60 NIC

61 NIC transmission processing part

62 Transmission buffer

63 NIC receiving processing part

64 Receive buffer

65 Receiving buffer register

66 Transmitting buffer register

70 1a 70 2 NIC device driver

71 171 2 transmission processing part

72 172 2 transmission buffer

73 173 2 receiving processing part

74 174 2 receive buffer

100 Computer

110 CPU

120 Input device  
130 ROM  
140 RAM  
150 Reader  
160 Output unit  
170 Bus  
200 Recording medium

---

## DESCRIPTION OF DRAWINGS

---

### [Brief Description of the Drawings]

[Drawing 1] It is a block diagram showing the outline composition of Embodiment 1 concerning this invention.

[Drawing 2] It is a block diagram showing the concrete composition of Embodiment 1.

[Drawing 3] It is a figure explaining memory mapping between the multi operating systems in Embodiment 1.

[Drawing 4] It is a figure explaining the principle of operation of Embodiment 1.

[Drawing 5] It is a figure explaining the principle of operation of Embodiment 1.

[Drawing 6] It is a figure explaining the example of Embodiment 1 of operation.

[Drawing 7] It is a flow chart explaining the example of Embodiment 1 of operation.

[Drawing 8] It is a flow chart explaining the example of Embodiment 1 of operation.

[Drawing 9] It is a block diagram showing the composition of Embodiment 2 concerning this invention.

[Drawing 10] It is a block diagram explaining the example of Embodiment 2 of operation.

[Drawing 11] It is a flow chart explaining the example of Embodiment 2 of operation.

[Drawing 12] It is a flow chart explaining the example of Embodiment 2 of

operation.

[Drawing 13] It is a figure explaining the applications 1-4 of Embodiment 1 or 2 concerning this invention.

[Drawing 14] It is a figure explaining the case where Embodiment 1 or 2 is applied to system shift.

[Drawing 15] It is a figure explaining the case where Embodiment 1 or 2 is applied to a high security gateway.

[Drawing 16] It is a figure explaining the case where Embodiment 1 or 2 is applied to a desktop Grid terminal.

[Drawing 17] It is a figure explaining the case where Embodiment 1 or 2 is applied to a remote management terminal.

[Drawing 18] It is a figure explaining the case where Embodiment 1 or 2 is applied to an efficient net service provision terminal.

[Drawing 19] It is a figure explaining the case where Embodiment 1 or 2 is applied to a high security Web Service offer server.

[Drawing 20] It is a block diagram showing the composition of the modification of Embodiment 1 or 2.

[Drawing 21] It is a block diagram showing the example 1 of composition of the conventional multi operating system.

[Drawing 22] It is a block diagram showing the example 2 of composition of the conventional multi operating system.

[Description of Notations]

10 Hardware

11 Control section

12 Physical memory

12a<sub>1</sub> the physical memory field for 12a<sub>2</sub>OS

12b<sub>1</sub> the physical memory field for 12b<sub>2</sub>AP

13 Interrupt vector table register

14 Page table register

15 Debugging register

16 Other registers

17 Program counter

18 The register for the data communications between OS's

20 <sub>1</sub>20 <sub>2</sub>OS

21 <sub>1</sub>21 <sub>2</sub> data check treating part

22 <sub>1</sub>22 <sub>2</sub> interruption distribution processing part

23 <sub>1</sub>23 <sub>2</sub> interrupt processing section

24 <sub>1</sub>24 <sub>2</sub> register data read-and-write treating part

25 <sub>1</sub>25 <sub>2</sub> interrupt vector table

26 <sub>1</sub>26 <sub>2</sub> page table

27 <sub>1</sub>27 <sub>2</sub> register save area

28 <sub>1</sub>a 28 <sub>2</sub>OS change treating part

29 <sub>1</sub>29 <sub>2</sub> data size judgment part

30 <sub>1</sub>30 <sub>2</sub>AP

40 <sub>1</sub>40 <sub>2</sub> virtual memory space

40a<sub>1</sub>virtual memory space for 40a<sub>2</sub>OS

40b<sub>1</sub>virtual memory space for 40b<sub>2</sub>AP

50 Network

60 NIC

61 NIC transmission processing part

62 Transmission buffer

63 NIC receiving processing part

64 Receive buffer

65 Receiving buffer register

66 Transmitting buffer register

70 <sub>1</sub>a 70 <sub>2</sub>NIC device driver

71 <sub>1</sub>71 <sub>2</sub> transmission processing part

72 <sub>1</sub>72 <sub>2</sub> transmission buffer

73 <sub>1</sub>73 <sub>2</sub> receiving processing part

74 <sub>1</sub>74 <sub>2</sub> receive buffer

100 Computer  
110 CPU  
120 Input device  
130 ROM  
140 RAM  
150 Reader  
160 Output unit  
170 Bus  
200 Recording medium

---